Lista 1 – Redes Neurais Artificiais

Kesley Roberto Ferreira Silva

Marcony Montini de Oliveira Lima

8EngCN16

1) Redes neurais são técnicas de tomadas de decisão e classificação baseadas no aprendizado de máquinas inspiradas no comportamento humano. As Redes neurais têm a vantagem do aprendizado a partir de métodos matemáticos. à medida que mais dados são inseridos, mais o erro do sistema se aproxima de zero de acordo com a técnica matemática definida.

2) Classificação de imagens, definição de rotas, identificação de caracteres manuscritos.

3) Um neurônio MCP é dado a partir do somatório do produto entre os valores das entradas pelos seus respectivos pesos. Em seguida, o valor passa por uma função de ativação que resulta em 1 caso o valor seja maior ou igual a zero e 0 caso contrário.

4) a) $y = g(10*0,8 -20*0,2 +4*(-1) -2*(-0,9)) = g(1,8) = 1,8$

b) $y = g(0,5 + 0*0,8 -20*0,2 +4*(-1) -2*(-0,9)) = g(2,3) = 1$

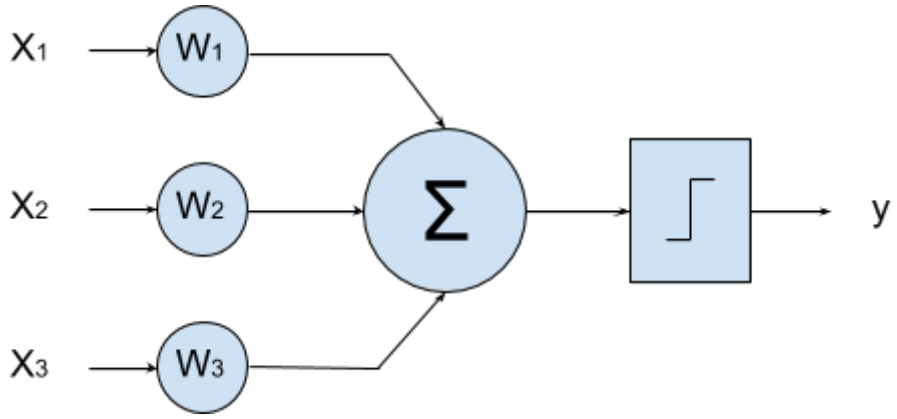
5) Plasticidade do sistema nervoso central é a capacidade que o cérebro possui de se adaptar a mudanças. Os neurônios se reorganizam criando novas estruturas e se moldando através dos aprendizados e vivências. As redes neurais possibilitam um comportamento semelhante através de seu treinamento. Assim, ao inserir um dado divergente do conjunto anterior, os pesos podem ser readequados para se aproximarem de um resultado final mais próximo do satisfatório.

6) Aprendizado supervisionado refere-se à forma de aprendizado de máquina em que a etapa de treinamento é feita de acordo com entradas e saídas já conhecidas.

Desta forma é possível encontrar relações matemáticas que produzem tais resultados. Exemplo: Regressão Linear e Classificação.

Aprendizado não supervisionado pelo contrário, utiliza apenas as entradas para a definição das saídas ou de um grupo de saídas através de semelhanças entre os dados. Exemplo: Agrupamento e Padrões sequenciais.

7) O desenho de um perceptron simplificado é mostrado a seguir.



O diagrama representa as entradas x1, x2 e x3, seus respectivos pesos w1, w2 e w3. O valor do produto entre as entradas e os pesos é acumulado em Σ e em seguida passa pela função de ativação degrau. Um perceptron deste tipo tem por característica a aplicação em sistemas linearmente separáveis. Como este fato não se aplica a uma função lógica XOR, este modelo não pode ser utilizado para tal.

8) Função Lógica OR, taxa de aprendizado de 0,5.

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

Começando com pesos iguais a zero. (w1 = w2 = 0)

y1 = g(0*0 + 0*0) = 0	erro = 0	erro total = 0	w1 = w2 = 0
y2 = g(0*0 + 1*0) = 0	erro = 1	erro total = 1	w1 = 0 + 0,5*0*1 = 0 w2 = 0 + 0,5*1*1 = 0,5
y3 = g(1*0 + 0*0,5) = 0	erro = 1	erro total = 2	w1 = 0 + 0,5*1*1 = 0,5 w2 = 0,5 + 0,5*0*1 = 0,5
y4 = g(1*0,5 + 1*0,5) = 1	erro = 0	erro total = 2	w1 = w2 = 0,5
y1 = g(0*0,5 + 0*0,5) = 0	erro = 0	erro total = 0	
y2 = g(0*0,5 + 1*0,5) = 1	erro = 0	erro total = 0	
y3 = g(1*0,5 + 0*0,5) = 1	erro = 0	erro total = 0	
y4 = g(1*0,5 + 1*0,5) = 1	erro = 0	erro total = 0	

Fim -> W1 = W2 = 0,5

9) a) Código de implementação de um Perceptron para a função OR:

```
import numpy as np

entradas = np.array([[0,0],[0,1], [1,0], [1,1]])
saidas = np.array([0,1,1,1])
pesos = np.array([0.0, 0.0])
taxaAprendizagem = 0.5

def stepFunction(soma):
    if (soma >= 1):
        return 1
    return 0

def calculaSaida(registro):
    s = registro.dot(pesos)
    return stepFunction(s)

def treinar():
    erroTotal = 1
    while (erroTotal != 0):
        erroTotal = 0
        for i in range(len(saidas)):
            saidaCalculada = calculaSaida(np.asarray(entradas[i]))
            erro = saidas[i] - saidaCalculada
            erroTotal += erro
            for j in range(len(pesos)):
                pesos[j] = pesos[j] + (taxaAprendizagem * entradas[i][j] * erro)
                print('Peso atualizado: ' + str(pesos[j]))
        print('Total de erros: ' + str(erroTotal))

treinar()
print('Rede neural treinada')
print(calculaSaida(entradas[0]))
print(calculaSaida(entradas[1]))
```

```
print(calculaSaida(entradas[2]))
print(calculaSaida(entradas[3]))
```

b) Código de implementação de um Perceptron para a função AND:

```
import numpy as np

entradas = np.array([[0,0],[0,1], [1,0], [1,1]])
saidas = np.array([0,0,0,1])
pesos = np.array([0.0, 0.0])
taxaAprendizagem = 0.5

def stepFunction(soma):
    if (soma >= 1):
        return 1
    return 0

def calculaSaida(registro):
    s = registro.dot(pesos)
    return stepFunction(s)

def treinar():
    erroTotal = 1
    while (erroTotal != 0):
        erroTotal = 0
        for i in range(len(saidas)):
            saidaCalculada = calculaSaida(np.asarray(entradas[i]))
            erro = saidas[i] - saidaCalculada
            erroTotal += erro
            for j in range(len(pesos)):
                pesos[j] = pesos[j] + (taxaAprendizagem * entradas[i][j] * erro)
                print('Peso atualizado: ' + str(pesos[j]))
        print('Total de erros: ' + str(erroTotal))

treinar()
print('Rede neural treinada')
print(calculaSaida(entradas[0]))
print(calculaSaida(entradas[1]))
print(calculaSaida(entradas[2]))
print(calculaSaida(entradas[3]))
```

10) Para o projeto de uma rede neural artificial deve-se levar em consideração os seguintes passos:

- Definição do problema.
- Escolha das informações:
    - Obtenção dos dados.

    - Criação de arquivos da rede.
- Treinamento da rede.
- Testes da rede.
- Adaptação para uso no problema definido.